



Instituto de Ciência e Tecnologia  
Universidade Federal de São Paulo

# Compiladores: Introdução

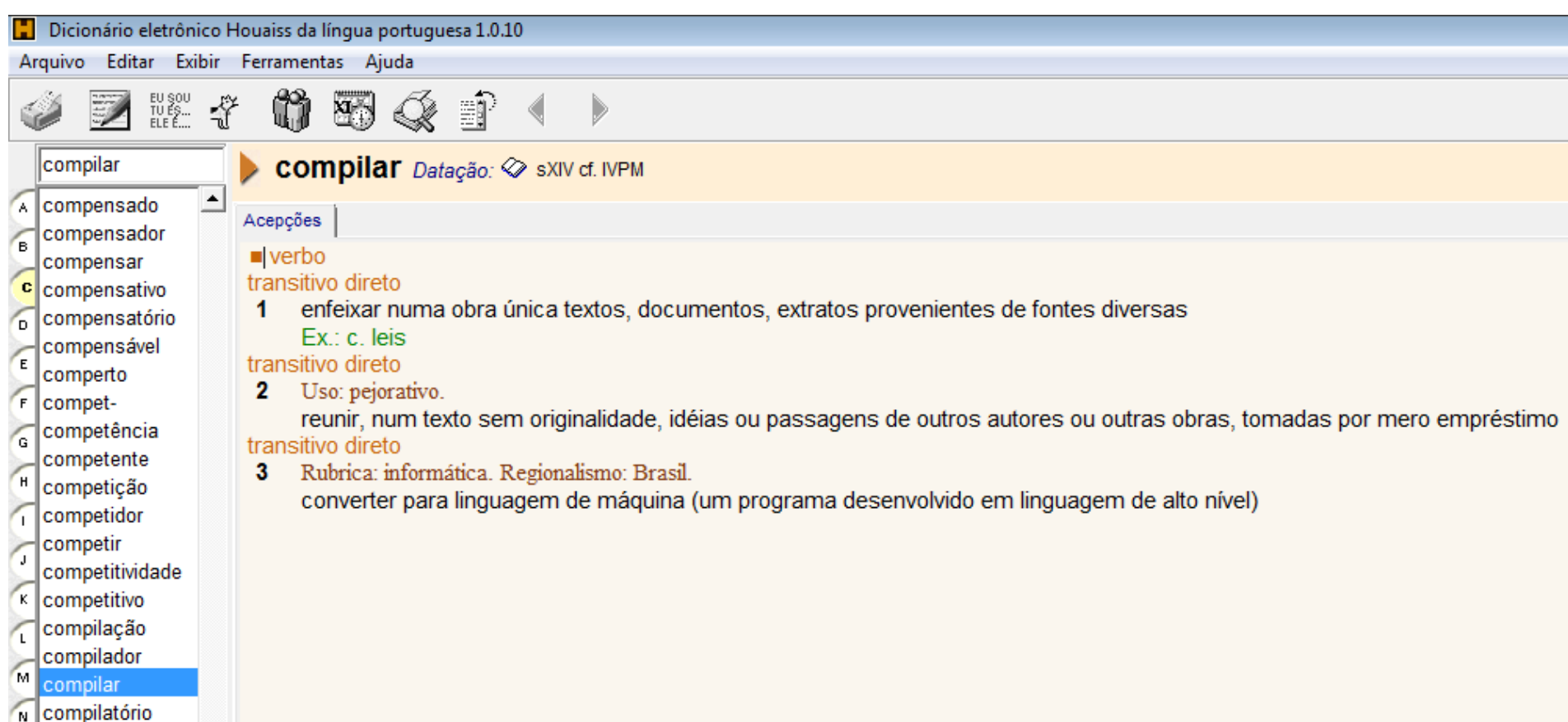
Prof<sup>a</sup> Thaína A. A. Tosta

[tosta.thaina@unifesp.br](mailto:tosta.thaina@unifesp.br)

São José dos Campos – 2021/2

# Introdução

O que significa a palavra **compilar**?



The screenshot shows the 'Dicionário eletrônico Houaiss da língua portuguesa 1.0.10' interface. The search bar contains the word 'compilar'. The left sidebar lists words from 'A' to 'N', with 'compilar' highlighted. The main panel displays the definition for 'compilar'.

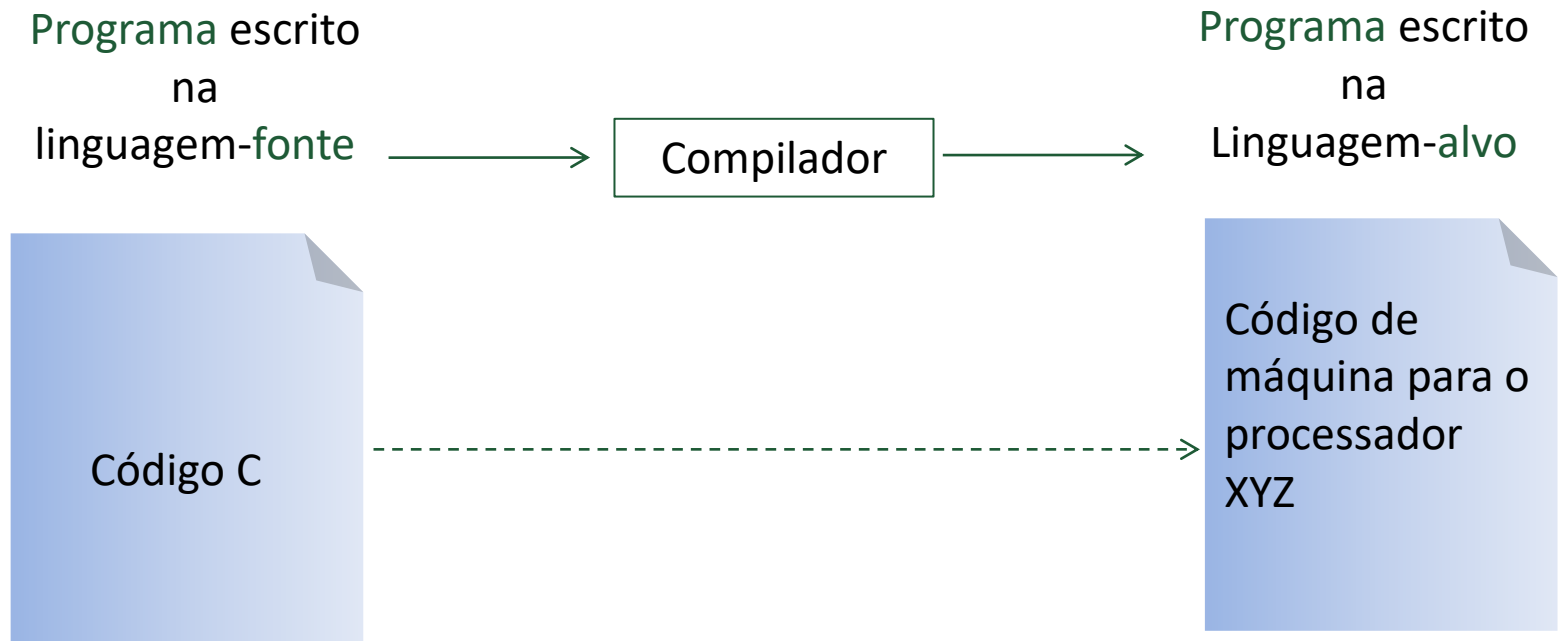
**compilar** *Datação:* sXIV cf. IVPM

**Acepções**

- **verbo**
  - transitivo direto**
    - 1 enfeixar numa obra única textos, documentos, extratos provenientes de fontes diversas  
*Ex.: c. leis*
    - transitivo direto**
      - 2 **Uso: pejorativo.**  
reunir, num texto sem originalidade, idéias ou passagens de outros autores ou outras obras, tomadas por mero empréstimo
      - transitivo direto**
        - 3 **Rubrica: informática. Regionalismo: Brasil.**  
converter para linguagem de máquina (um programa desenvolvido em linguagem de alto nível)

# Introdução

Um compilador é um programa de computador que **traduz** uma linguagem para outra.



# Introdução

## Por que estudar compiladores?

- Os compiladores são usados em várias formas de computação;
- Os compiladores têm incorporado algoritmos cada vez mais sofisticados;
- Além disso, também tem ocorrido o desenvolvimento de linguagens de programação cada vez mais sofisticadas.

# Introdução

## Compiladores são realmente importantes?

Sem eles, ainda estaríamos na “idade da pedra” do desenvolvimento de *software*, escrevendo programas por linguagem de máquina.



Ex: processadores 8x86 – IBM PC

C7 06 0000 0002 (código hexadecimal – **linguagem de máquina**)

MOV x , 2 (código de montagem – **linguagem de montagem**)

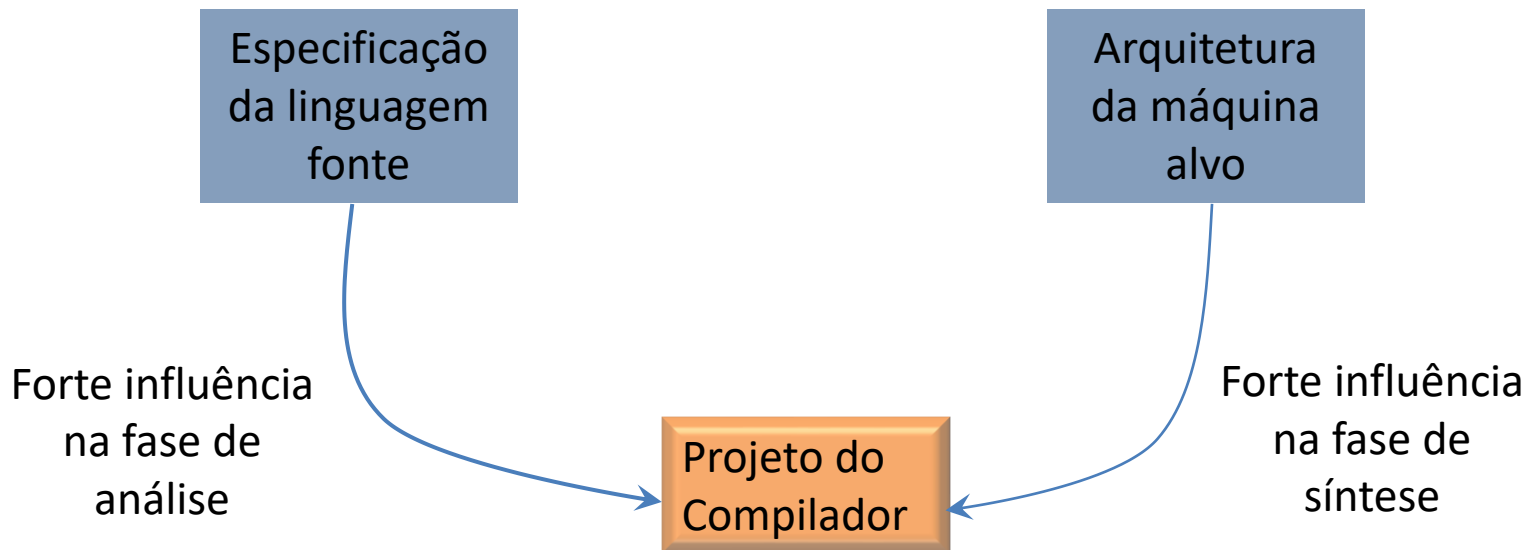
x = 2 (código em **linguagem de alto nível**)



montador

# Introdução

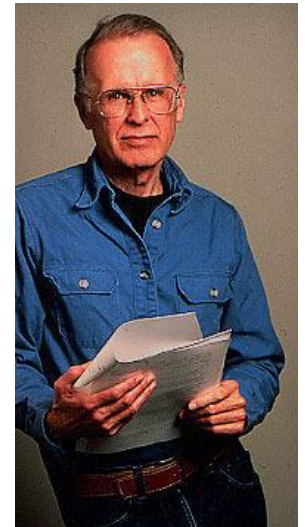
## Principais fatores que influenciam o projeto de um compilador:



# Introdução

## Histórico:

- 1952: primeiro compilador por Grace Hopper, para a linguagem *A-0 System*;
- 1957: primeiro compilador completo (projeto liderado por John W. Backus, na IBM), para a linguagem **FORTRAN**;
- 1960: surgimento de compiladores para múltiplas plataformas, inicialmente para a linguagem **COBOL**;
- 1962: primeiro *self-hosting compiler* (escrito com a própria linguagem que o compilador traduz), para a linguagem **LISP**;
- Da década de 70 em diante tornou-se comum implementar compiladores com a própria linguagem que ele compila.



# Introdução

## Programas relacionados a compiladores:

- Interpretador

Interpreta o código-fonte e executa-o imediatamente, não gera código-objeto;

- Montador (*assembler*)

Traduz a linguagem de montagem (*assembly*) de um processador particular;

- Organizador (*linker*)

Coleta o código compilado separadamente e coloca tudo em um arquivo executável.



# Introdução

## Programas relacionados a compiladores:

- Carregador (*loader*)

Carrega o executável na memória, resolve os endereços relocáveis relativos a um dado endereço base ou inicial;

- Pré-processador

Ativado pelo compilador antes do início da tradução, pode apagar comentários, incluir outros arquivos e executar substituições de macros;

- Editor

Oferece infraestrutura para escrever o programa fonte, gerando o arquivo a ser compilado, pode ser orientado pela estrutura ou formato da linguagem.

# Introdução

## Programas relacionados a compiladores:

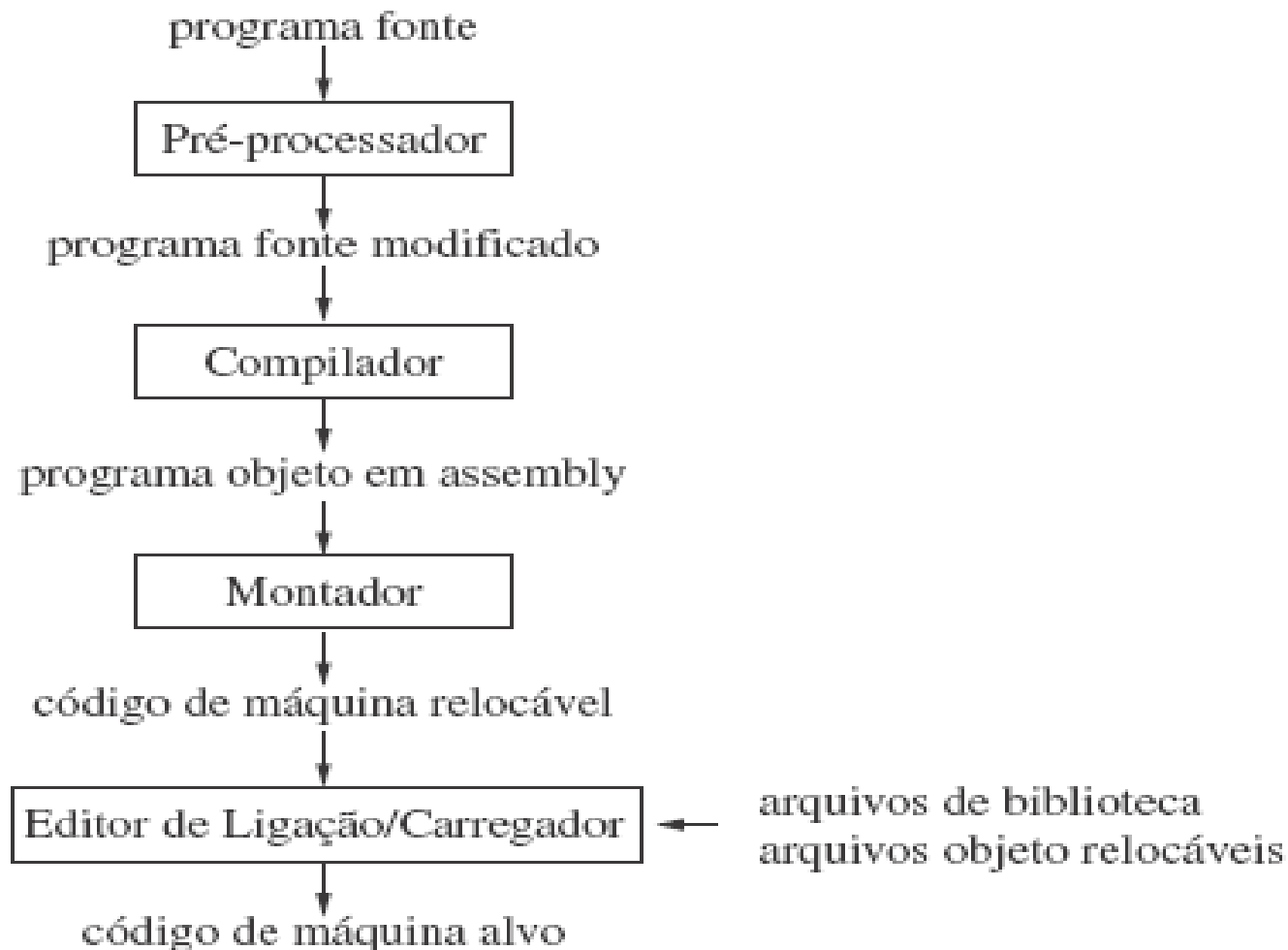
- Depurador

Utilizado para determinar erros de execução em um programa compilado, costuma ser utilizado de forma integrada em um IDE (*Integrated Development Environment*);

- Gerador de perfil

Coleta estatísticas sobre o comportamento de um programa objeto durante sua execução.

# Introdução



**FIGURA 1.5** Um sistema de processamento de linguagem.

# Introdução

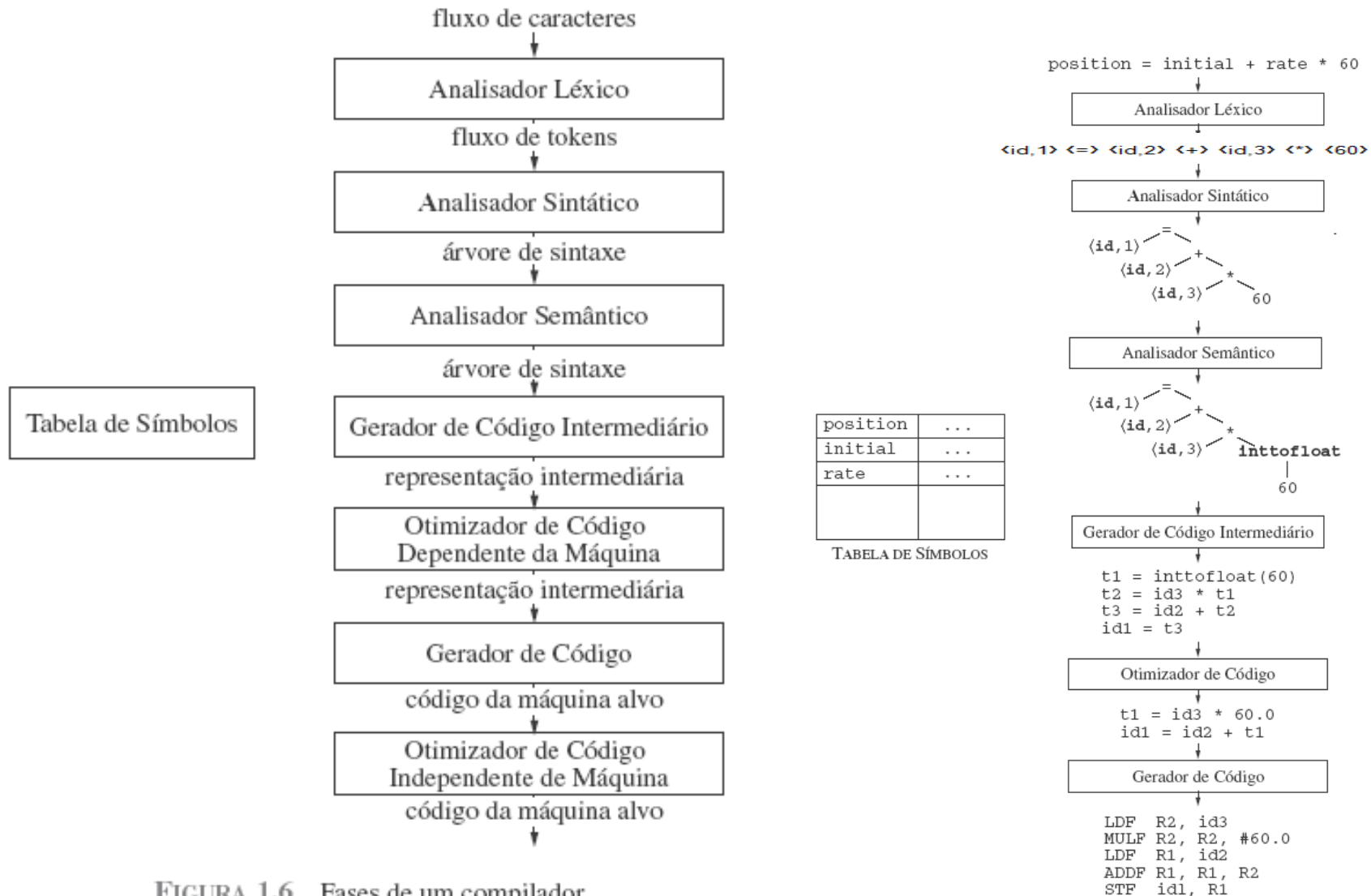


FIGURA 1.6 Fases de um compilador.

# Introdução

- Análise Léxica (analisador léxico – sistema de varredura)  
Sequências de caracteres são organizadas como unidades significativas, chamadas lexemas.

Exemplo:  $a[\text{index}] = 4 + 2$

Lexemas	<i>Tokens</i>
a	Identificador
[	Colchete à esquerda
index	Identificador
]	Colchete à direita
=	Atribuição
4	Número
+	Adição
2	Número

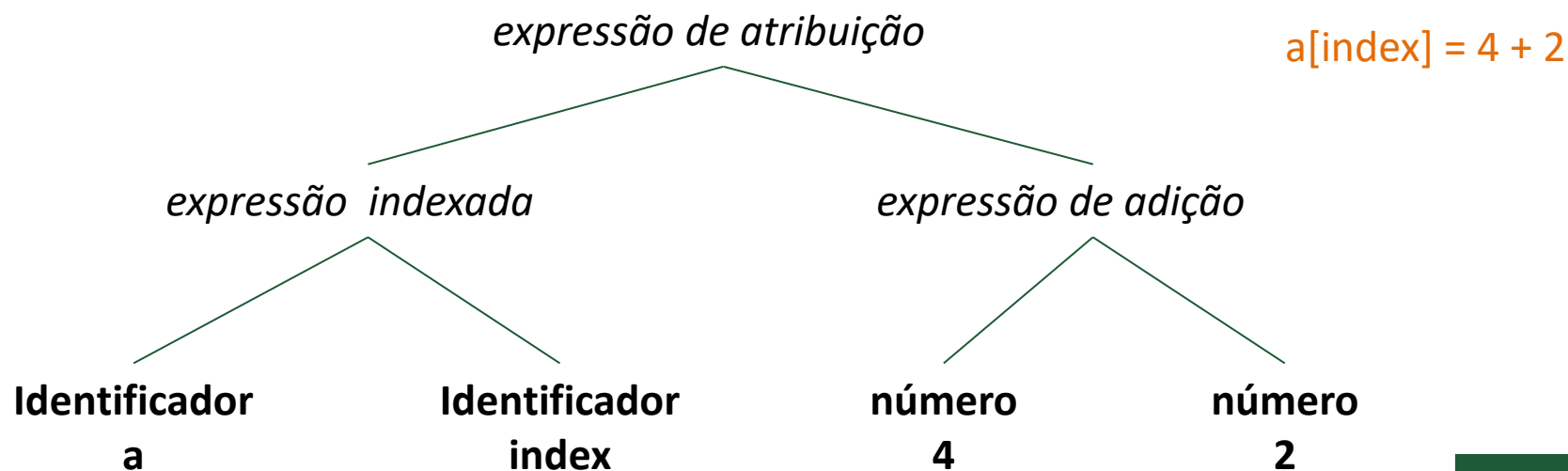
Além de reconhecer os *tokens*, o analisador léxico pode inserir identificadores na tabela de símbolos.

# Introdução

- Análise Sintática (*analisador sintático – parser*)

Determina os elementos estruturais do programa e seus relacionamentos;

Os resultados da análise sintática geralmente são representados como uma *árvore de análise sintática* ou uma *árvore sintática*.

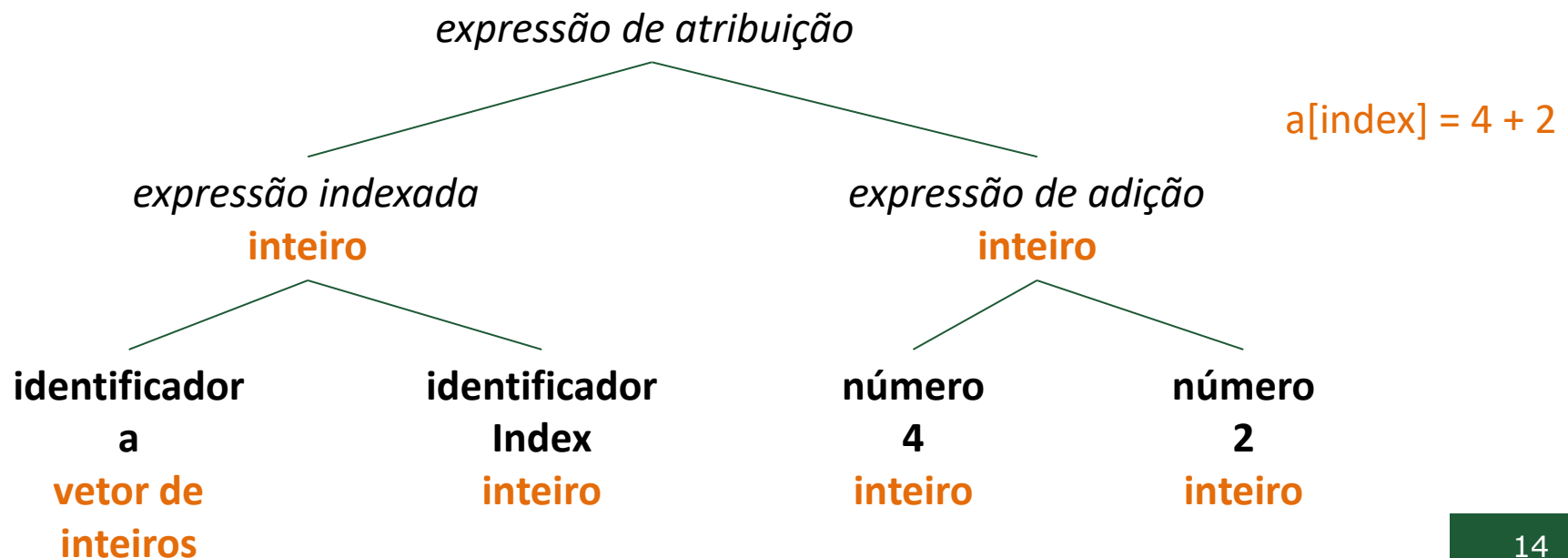


# Introdução

- Análise Semântica (**analisador semântico**)

A semântica de um programa é seu significado;

O analisador semântico faz a verificação de tipos e declarações (semântica estática), e frequentemente adiciona novas informações (atributos) na árvore sintática.



# Introdução

- A fase de **análise**

<b>Análise léxica</b>	Verifica se a <b>palavra</b> está bem formada
<b>Análise sintática</b>	Verifica se a <b>sentença</b> está bem formada
<b>Análise semântica</b>	Verifica se o <b>texto</b> (análise de tipos) está coerente



# Introdução

- Gerador de Código Intermediário

Gera uma representação intermediária linearizada, próxima do código de montagem que deve ser facilmente produzida e traduzida para a máquina alvo;

Uma representação intermediária muito utilizada é o **código de três endereços**

Exemplo:  $a[\text{index}] = 4 + 2$



$t = 4 + 2$

$a[\text{index}] = t$

# Introdução

- Otimizador de Código Intermediário

Transforma o código intermediário com o objetivo de produzir um código objeto melhor;

Código objeto melhor pode significar: um código mais rápido, menor ou que consuma menos energia.

Exemplo: o código de três endereços abaixo

$t = 4 + 2$

$a[index] = t$

pode ser otimizado para

$t = 6$  (empacotamento constante)

$a[index] = t$

e depois para

$a[index] = 6$

# Introdução

- Gerador de Código Alvo

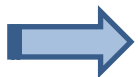
A partir do código intermediário otimizado, gera o código para a máquina alvo;

Nessa fase as propriedades da máquina alvo se tornam o fator principal:

- Conjunto de instruções;
- Modos de representação de dados;
- Modos de endereçamento;
- Conjunto de registradores.

Exemplo:  $a[index] = 6$

Linguagem de  
montagem  
hipotética



```
MOV R0, index    ;; move valor de index para R0
MUL R0, 2         ;; dobra o valor em R0 (inteiro ocupando 2 bytes)
MOV R1, &a        ;; move endereço de a para R1
ADD R1, R0        ;; adiciona R0 a R1
MOV *R1, 6        ;; move o valor 6 para o endereço apontado em R1
```

# Introdução

- Otimizador de Código Alvo

O compilador tenta melhorar o código alvo gerado, como:

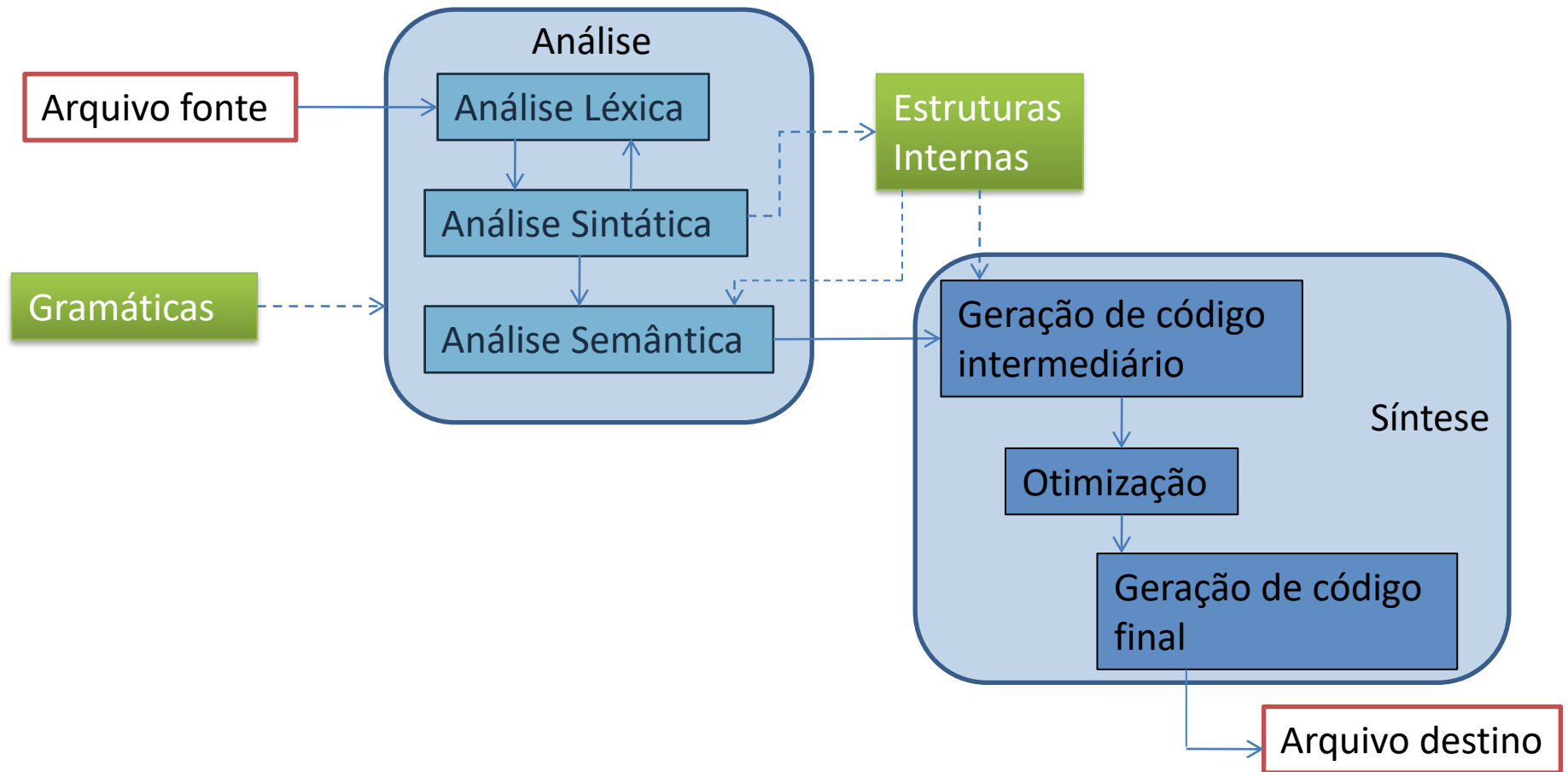
- Escolher outros modos de endereçamento (melhora desempenho);
- Substituições de instruções lentas por outras mais rápidas (eliminação de operações redundantes ou desnecessárias).

Exemplo:  $a[\text{index}] = 6$

MOV R0, index	MOV R0, index	;; move valor de index para R0
MUL R0, 2	SHL R0	;; instrução de deslocamento (dobra valor em R0)
MOV R1, &a	MOV &a[R0], 6	;; endereçamento indexado
ADD R1, R0		
MOV *R1, 6		

# Introdução

- Diferentes visões do processo de tradução:

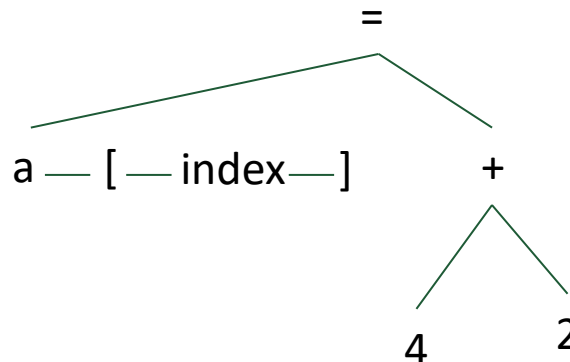


# Introdução

- Diferentes visões do processo de tradução:



$a[\text{index}] = 4 + 2$



OU  
 $t = 4 + 2$   
 $a[\text{index}] = t$

```
MOV R0, index
MUL R0, 2
MOV R1, &a
ADD R1, R0
MOV *R1, 6
```

# Introdução

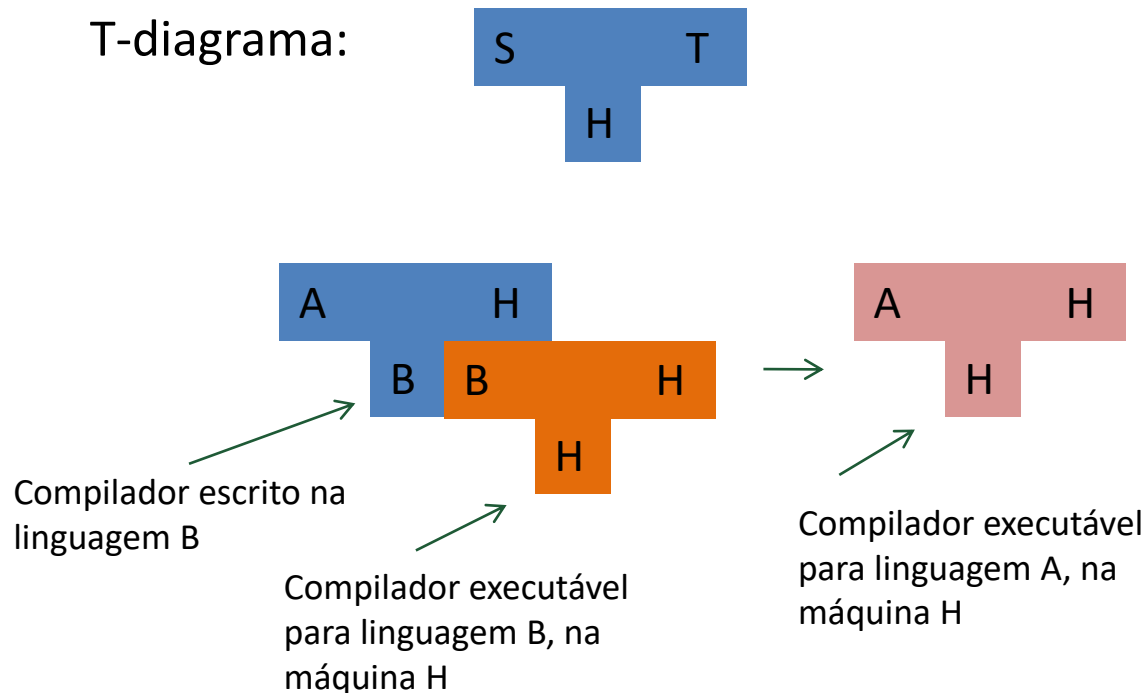
## Passadas:

- É comum um compilador processar um programa fonte diversas vezes, com repetições chamadas de **passadas**;
- A passada inicial constrói a árvore sintática ou o código intermediário;
- As demais passadas consistem em acrescentar informações, alterando a estrutura ou produzindo uma representação diferente.

# Introdução

## Como criamos o compilador do compilador ?

**Abordagem 1:** escrever o compilador em linguagem diferente da qual ele deve compilar (para qual já exista um compilador executável)



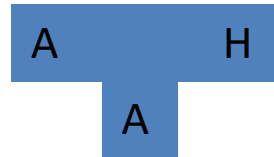


# Introdução

## Como criamos o compilador do compilador?

**Abordagem 2:** escrever o compilador na mesma linguagem que ele deve compilar (para qual não exista um compilador executável)

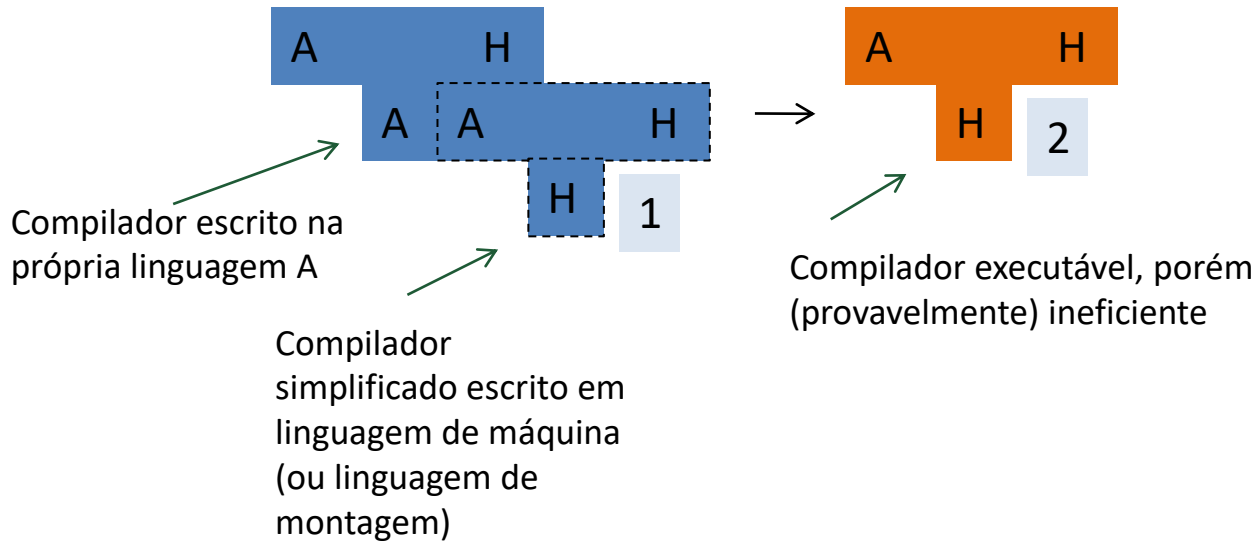
Exemplo:



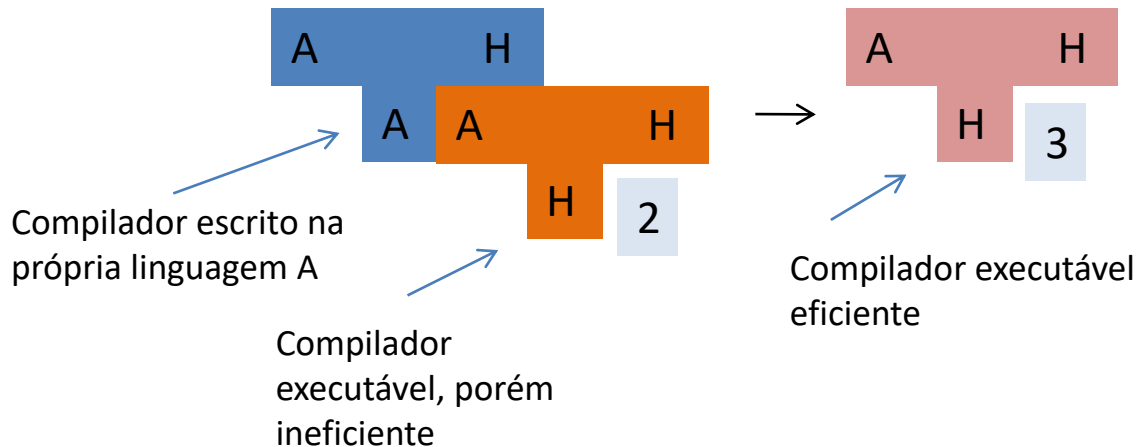
Processo de **partida rápida** (*bootstrapping*)

# Introdução

## Partida Rápida (*Bootstrapping*):



*Primeiro passo do processo de partida rápida*



*Segundo passo do processo de partida rápida*

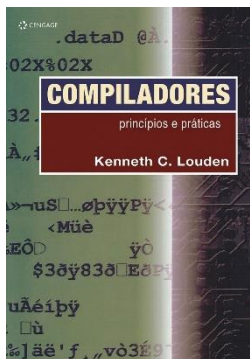
# Introdução

## Vantagens da Partida Rápida (*Bootstrapping*):

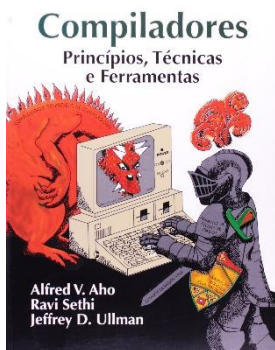
- Testar a capacidade da linguagem fonte do compilador (linguagem nova criada);
- Facilitar a transposição do compilador para outro computador hospedeiro, requerendo apenas reescrever o módulo de síntese do código-fonte do compilador.

# Introdução

## Bibliografia consultada:



LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004 (Cap. 1);



AHO, A. V.; LAM, M. S.; SETHI, R. e ULLMAN, J. D. **Compiladores: princípios, técnicas e ferramentas.** 2ª edição – São Paulo: Pearson Addison-Wesley, 2008 (Cap. 1).